FORMALIZACIÓN DE PATRONES DE DISEÑO DE COMPORTAMIENTO

A.Cortez, A. Garis, D. Riesco.

Departamento de Informática - Facultad de Ciencias Físico-Matemáticas y Naturales. UNSL.
Instituto de Informática . Facultad de Ingeniería. Universidad de Mendoza.
cortezalberto@gmail.com, agaris. driesco@unsl.edu.ar, driesco@unsl.edu.ar

Resumen

Los patrones de diseño son una herramienta muy útil de la ingeniería de software. Un significativo aporte es el que brindan los denominados “patrones de comportamiento” (clasificación Gof [3]) , ya que facilitan el manejo de comportamientos complejos. Uno de los requerimientos para la implementación exitosa de un patrón es que posea una especificación clara El concepto de Perfiles UML ha sido empleado en la especificación de patrones de diseño de tipo estructural (según clasificación Gof). El presente trabajo propone un camino para la especificación de patrones de comportamiento haciendo uso de esta metodología.

Palabras clave: patrones de comportamiento, perfiles UML, especificaciones.

Contexto

La línea de investigación planteada se desarrolla en el marco del proyecto 22/F822 "Ingeniería de Software: Conceptos, Métodos y Herramientas en un contexto de Ingeniería de Software en Evolución”, Facultad de Ciencias Físico-Matemáticas y Naturales. UNSL**.** (Universidad Nacional de San Luis).

1. Introducción

El lenguaje de modelado UML es uno de los estándares más utilizado para especificar y documentar sistemas. Y ha sido utilizado para especificar patrones de diseño. Pero ocurre que a veces no es suficientemente formal para lograr una especificación precisa. Los Perfiles UML constituyen el mecanismo que proporciona el propio UML para extender su sintaxis y su semántica de manera de expresar los conceptos específicos de un determinado dominio de aplicación. Los perfiles no solo pueden ser utilizados para dominios específicos, sino también para resolver problemas particulares en diferentes dominios. El avance en el uso de perfiles para definir patrones es un instrumento de gran utilidad tanto a nivel conceptual como a nivel práctico. A nivel conceptual se permite definir, visualizar y documentar los patrones de diseño. A nivel práctico se propicia la definición de los patrones de manera estándar. De esta manera no es necesario crear una nueva herramienta para especificar patrones,

sino que es posible usar las herramientas de UML existentes [1]. Esencialmente, es posible realizar la validación con diagramas de clases UML que posean las características propias de un patrón de diseño determinado.
Se utilizan dos herramientas: UML (Universal Modeling Language) y OCL (Object Constraint Language) [10][11]. El mecanismo aplicado consiste en definir un perfil UML por cada patrón al que se le anexan restricciones definidas en OCL. En este trabajo se recurre a una arquitectura de patrones, compuesta de una jerarquía integrada por: un nivel general, un nivel intermedio para la clasificación Gof y otro inferior para cada tipo particular de patrón.

2. Líneas de investigación y desarrollo

La línea de investigación aquí presentada se refiere al uso de Perfiles UML para la especificación de patrones de comportamiento. Se explica a continuación las bases y la metodología utilizada para su elaboración.
Para la definición de patrones se trabajó sobre una arquitectura de patrones, compuesta por tres niveles [5] . Cada uno de los niveles contiene especificaciones. Las especificaciones del nivel 0 son generales para todos los patrones mientras que las especificaciones del nivel 1 incluyen perfiles particulares para patrones de clase, de objeto, estructurales, creacionales y de comportamiento. El nivel 2 contiene las especificaciones de un determinado patrón. El objetivo del presente trabajo es enriquecer la arquitectura en el nivel 1 con definiciones propias de los patrones de comportamiento. Y definir las especificaciones de todos los patrones de comportamiento del nivel 2.

A continuación se definen los pasos de la metodología utilizada:

1. Identificar los participantes principales del patrón: clases y objetos participantes del patrón junto con sus responsabilidades. En este punto se trata de simplificar la cantidad de elementos que componen el patrón.
2. Definir un estereotipo por cada participante del patrón.
3. Incluir, si es necesario, un valor etiquetado asociándolo a un estereotipo.
4. Definir las restricciones que debe cumplir el patrón a través de restricciones sobre los estereotipos, representado el comportamiento deseado para el mismo. En este punto se consideran las restricciones existentes en el nivel 0 y 1, pudiéndose agregar nuevas restricciones para especificar el nuevo patrón.

3. Resultados y Objetivos

El presente trabajo tiene como objetivo lograr la especificación de todos los patrones de diseño clasificados como de comportamiento según Gof. Se trabajó en primer término de simplificar los modelos definidos por Gof para cada patrón. Una vez refinados los modelos se comenzó con la elaboración de especificaciones para cada uno de los patrones. Valiéndose de la metodología explicada en la sección anterior se han especificado diferentes patrones de comportamiento Gof. Como ejemplo de ello se puede mostrar lo logrado con el patrón Command. En la figura 1 se puede observar el perfil logrado que contiene cuatro estereotipos: Command, ConcreteCommand, Invoker y execute.

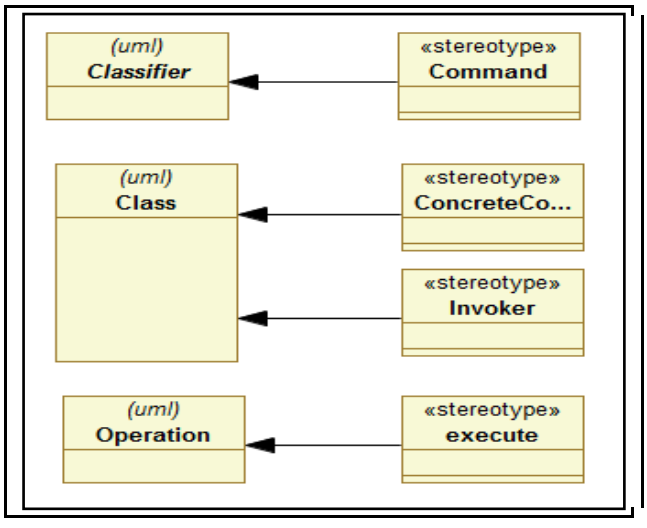

Figura 1 Perfil del patrón Command

Para el estereotipo execute se definió la siguiente especificación:

-- "execute" debe ser redefinido en la clase estereotipada "ConcreteCommand"

Context Operation inv:
IsStereotypedBy (self,Command) implies implementedOperation (self, ConcreteCommand)

implementedOperation() es una restricción definida en el nivel 0 de la arquitectura y sirve como base para la construcción de las restricciones del nivel 2.

context DesignPatternFrameworkProfile::
implementedOperation(o:Operation,c:String): Boolean
implementedOperation = Class.allInstances
->exists(c1| isStereotypedBy(c1,s) and
cl.allMethods->exists(m| o.name=m.name and m.specification->notEmpty()
implies (not(m.specification.isAbstract) and m.specification.ownedParameter
-> includesAll(o.ownedParameter))))

implementedOperation(o: Operation, s: Stereotype) : devuelve true si la operación "o" es implementada en una clase estereotipada con "s", con un método cuyo nombre es el mismo de "o"; false en otro caso.

De esta forma se avanza en la arquitectura para el resto de los patrones. Se espera mejorar las especificaciones definidas hasta el momento y además han surgido proyectos asociados. Por lo que se espera poder refinar de tal modo las especificaciones que sirvan como base para desarrollos futuros. Entre ellos se quiere obtener:

- La definición de una arquitectura para especificación de antipatrones.
- Un modelo para la detección de los elementos de un diagrama que coinciden con un patrón. La propuesta aquí presentada, facilita esta tarea ya que aplicando los estereotipos en los diagramas y validando las restricciones se determina si coincide o no.
- Un framework para validar un modelo asociado a un patrón.

Dado que el campo de trabajo es amplio, las necesidades y posibilidades de mejorar y colaborar en el desarrollo de modelos crecen junto con las necesidades del mercado.

4. Formación de recursos humanos

En el marco de este proyecto se está desarrollando una tesis de maestría en la Facultad de Ciencias Físico-Matemáticas y Naturales de la Universidad de San Luis.

Este trabajo también se vincula al Instituto de Informática de la Facultad de Ingeniería de la Universidad de Mendoza aportando conocimiento a la línea de investigación " Ingeniería de Software. Herramientas automáticas de código en el contexto de Ingeniería Dirigida por Modelos (MDE)"

5. Referencias

[1] Barotto V., Demonte M., Riesco D. 2005. Definición de Patrones de Diseño a través del metamodelo UML. Tesis de Licenciatura en Ciencias de la Computación, Universidad Nacional de Río IV, Argentina.

[2] Debnath N., Garis A., Riesco D., Montejano G. 2006. Defining Patterns using UML Profiles. *ACS/IEEE International Conference on Computer Systems and Applications*, IEEE Press, www.ieee.org, pp.1147-1150.

[3] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John 1994. Design Patterns: Elements of Reusable software, Addison-Wesley.

[4] García Carlos Diego. 2009. Tesis "Implementación de técnicas de evaluación y refinamiento para OCL 2.0 sobre múltiples lenguajes basados en MOF".

[5] Garis, Ana.Perfiles UML para la definición de Patrones de Diseño 2007.Tesis de maestría. Universidad Nacional de San Luis (UNSL).

[6] Gogolla, Martin, Bohling, Jorn and Richters, Mark 2003.Validation of UML and OCL Models by Automatic Snapshot Generation". Proc. 6th Int. Conf. Unified Modeling Language (UML'2003). Springer, Berlin, pages 265-279.

[7] MDA 2007. Model-Driven Arquitecture. Disponible en www.omg.org/mda.

[8] MOF 2006.Meta Object Facility. Documento: formal. Disponible en www.omg.org/mof.

[9] Martinez, Liliana 2008.Componentes MDA para patrones de diseño. Tesis de Maestría. Universidad Nacional de La Plata.

[10]OMG Superstructure 2010.Standard document URL: http://www.omg.org/spec/UML/2.3/Superstructure

[11] UML Profile for Patterns Specification 2010. Disponible en http://www.omg.org/spec/UML/2.3/.